Pergamon

Computers Math. Applic. Vol. 27, No. 5, pp. 63–70, 1994

0898-1221/94 $6.00 + 0.00

0898-1221(94)E0006-6

An Authentication-Combined Access Control Scheme Using a One-Way Function

T.-C. WU
Department of Information Management
National Taiwan Institute of Technology
Taipei, Taiwan 107, R.O.C.

C.-C. CHANG
Institute of Computer Science and Information Engineering
National Chung Cheng University, Chiayi, Taiwan 601, R.O.C.

Y.-S. YEH
Institute of Computer Science and Information Engineering
National Chiao Tung University, Hsinchu, Taiwan 300, R.O.C.

(Received November 1992; accepted June 1993)

Abstract—In this paper, we propose an authentication-combined access control scheme for information protection systems. Let a_{ij} be the access privilege of User i to File j. Initially, by the Diffie-Hellman public key distribution scheme, the system and the users are assigned distinct secret keys, and their corresponding public keys, respectively. Let K_s be the secret key and y_s be the public key of the system, and let K_i be the secret key and y_i be the public key of the User i. By using a predefined one-way function F, we compute $r_{ij} = F(K_i, y_s, a_{ij})$. Reversely, the access privilege can be retained as $a_{ij} = F(K_s, y_i, r_{ij})$. Being different from the previously proposed schemes, our scheme is safer and the user's secret key is used not only for computing the corresponding access privilege to the intended file, but also for authenticating the requesting user not to illegitimately access the protected files. The proposed scheme is simple to establish. Besides, it can perform the access control in dynamic environments, such as change access privileges and insert/delete users or files.

1. INTRODUCTION

With the advent of computer systems, vast amounts of personal, financial, commercial, and technological information are stored in computer data banks and shared by various users. In order to ensure the privacy and data security of this information, the necessity for providing access control mechanisms in information systems has become overwhelming [1,2]. In 1972, Graham and Denning introduced an abstract protection model for access control [2]. In their model, an information protection system basically consists of the following elements:

(1) a set of subjects,
(2) a set of objects, and
(3) an access control matrix.

Each entry (i, j) of the access control matrix represents the access privilege of subject i to object j. In practical applications, a user can be regarded as a subject and a file can be regarded as an object. Figure 1 shows an access control matrix with four users and five files. In Figure 1, we see that User 1 has "no access" to File 5, User 2 has "write" privilege to Files 1 and 2, and so on.

Users i \ Files j	1	2	3	4	5
1	4	4	1	2	0
2	2	2	1	0	3
3	0	1	4	3	3
4	1	2	0	0	4

0 : no access
1 : read
2 : write
3 : execute
4 : owned

Figure 1. An access control matrix with four users and five files.

Conventionally, three methods are used to implement the access control matrix, namely, as the accessor-list method [3], the capability-list method [3], and the key-lock matching method [2]. Although the accessor-list and the capability-list methods are easy to be constructed by linked lists, both methods have disadvantages, such as propagation, revocation, and review problems. As to the key-lock matching method, it suffers from finding such a key-lock pair.

In the past decade, several schemes have been proposed for the implementation of access control matrices [4–10]. Among them, the information protection system was established through sophisticated computation. Further, the whole system may need to be reestablished for the access control in dynamic environments, such as change access privileges and insert/delete users or files. Another common drawback exhibited by the previously proposed schemes is that the constructed system cannot withstand potential attacks, such that an intruder may pretend to be a certain legal user and enter the system to illegitimately access protected files. And this insecure leak will make the access control efforts be in vain.

The aim of this paper is to propose an authentication-combined access control scheme to overcome the disadvantages stated above. Before describing our scheme, we will give brief reviews of previous research works on access control.

2. PREVIOUS RESEARCH WORKS ON ACCESS CONTROL

Suppose there are m users and n files in the information system. For simplicity, let i be the identification number of User i, j be the identification number of File j. Let a_{ij} be the access privilege of User i to File j. Wu and Hwang [10] first initiated an elegant single-key-lock scheme for implementing an access control matrix. In their scheme, a random $m \times m$ nonsingular matrix is chosen. The i^{th} row vector in that matrix is assigned as the key $\mathbf{K}_i$ of User i. By solving a system of linear equations in a Galois field GF(p), an m-dimensional vector $\mathbf{L}_j$, used as the lock of File j, is computed from

$$a_{ij} = \mathbf{K}_i * \mathbf{L}_j \bmod p, \tag{2.1}$$

where the operator $*$ means the inner product in GF(p), and p is a prime number greater than the maximum value of the a_{ij}'s. Once the keys and locks are determined, it is easy to recompute the access privilege from (2.1). The concept of single-key-lock is very simple. However, the construction of $\mathbf{L}_j$'s is computationally sophisticated and time-consuming when the number of users is large. Recently, Chang and Jiang [7] proposed a binary implementation of a single-key-lock system to improve the computation time of the $\mathbf{L}_j$'s. However, by Wu and Hwang's method and Chang and Jiang's method, changing access privileges and inserting/deleting users or files will lead the whole system to be reestablished.

Later, Chang [4] proposed a key-lock-pair scheme based on the Chinese remainder theorem. In his method, a set of relatively prime numbers $L_1, L_2, \ldots, L_n$ are assigned as the locks of files $1, 2, \ldots, n$, respectively. Then the key K_i of User i is computed as

$$K_i = \sum_{j=1}^{n} D_j \, b_j \, a_{ij}, \qquad \text{where } D_j = \prod_{\substack{k=1 \\ k \neq j}}^{n} L_k \text{ and } D_j \, b_j \equiv 1 \pmod{L_j}. \tag{2.2}$$

The access privilege is computed as

$$a_{ij} = K_i \bmod L_j. \tag{2.3}$$

Another key-lock-pair scheme was also proposed by Chang [5] based upon Euler's theorem. Again, the locks L_j's are assigned relatively prime numbers. The key K_i of User i is computed as

$$K_i = \sum_{j=1}^{n} \left(N_{ij}\, n \prod_{\substack{k=1 \\ k \neq j}}^{n} L_k^{\phi(L_k)} \right), \tag{2.4}$$

where $N_{ij} = \lceil a_{ij}\, L_j / n \rceil$ and ϕ is Euler's totient function. The access privilege is computed as

$$a_{ij} = \left\lfloor \frac{K_i}{L_j} \right\rfloor \bmod n. \tag{2.5}$$

From the above two methods proposed by Chang, we see that K_i needs to be recomputed when an access privilege a_{ij} is changed. To insert a user, the corresponding key K_i is computed. As to insert/delete a file, all K_i's need to be recomputed.

Recently, two single key schemes for access control were proposed by Jan [8] and Laih, Harn, and Lee [9], individually. In Jan's method, the key K_i of User i is computed as

$$K_i = \sum_{j=1}^{n} a_{ij}\, (t+1)^{j-1}, \tag{2.6}$$

where t is the maximum value of the a_{ij}'s; and the access privilege is computed as

$$a_{ij} = \left\lfloor \frac{K_i}{(t+1)^{j-1}} \right\rfloor \bmod (t+1). \tag{2.7}$$

In Laih, Harn, and Lee's method, the keys of users are selected such that $K_i \neq K_j$ and $K_i \neq K_j \bmod p$, where p is a prime number greater than the maximum value of the a_{ij}'s. By using Newton's interpolation method [11], the access privilege is computed as

$$\begin{aligned} a_{ij} &= G_{nj}\,(K_i - K_{n-1})\,(K_i - K_{n-2}) \ldots (K_i - K_1) + \cdots + G_{2j}\,(K_i - K_1) + G_{1j} \pmod p, \\ \text{where} \quad G_{ij} &= \frac{\sum_{t=1}^{i-1} G_{tj} \prod_{s=1}^{t-1} (K_i - K_s)}{\prod_{s=1}^{n-1} (K_n - K_s)} \pmod p \qquad \text{and } G_{1j} = a_{1j}. \end{aligned} \tag{2.8}$$

It can be seen that Jan's method has the overflow problem when computing the keys if the number of files is large, and Laih, Harn, and Lee's method suffers from the computational complexities for solving the G_{ij}'s. Derived from Jan's method, another single key scheme for inserting new users and files has been proposed by Chang and Jan [6]. By the recursive characteristic of Newton's interpolation method, it is easy to insert users or files in Laih, Harn, and Lee's method. When deleting a file, all K_i's should be recomputed by Jan's method; in Laih, Harn, and Lee's method, deleting User i needs to recompute K_j's for $j > i$.

Note that the keys (or locks) produced from the above methods are managed by the system. Consequently, an intruder may pretend to be a certain legal user and illegitimately access the protected files.

3. AUTHENTICATION-COMBINED ACCESS CONTROL

3.1. One-Way Function and Key Distribution

One-way functions are significantly useful in solving cryptographic problems [12]. By a one-way function, we mean a function F such that

(1) $F(x)$ is easy to compute for any given x in the domain of F,
(2) given any y such that $y = F(x)$ for some x, it is computationally infeasible to find x unless certain special information used in the design of F is known.

The Diffie-Hellman public key distribution scheme [13] is one of the well known one-way functions based on the computing discrete logarithms problem. We first describe the Diffie-Hellman public key distribution scheme in the following.

Let p be a large prime number and α be a primitive element $\bmod p$. Let K be the secret key. The public key y corresponding to K is computed as

$$y = \alpha^K \bmod p. \tag{3.1}$$

From (3.1), it is easy to compute y for a given K; while computing K from y is equivalent to the problem of computing discrete logarithm in a Galois field GF(p). Further, if K is between 0 and $p-1$, then the relation of K to y is a one-to-one correspondence. That is, y is uniquely determined for a given K between 0 and $p-1$, or vice versa. It was pointed out [14] that if we carefully choose a prime number p such that $(p-1)/2$ is also a prime, then computing the discrete logarithm in GF(p) is very difficult.

The Diffie-Hellman public key distribution scheme can be used for constructing a common key shared by two communicating members. Let K_a be the secret key and y_a be the public key of User A, and, similarly, let K_b be the secret key and y_b be the public key of User B, derived from (3.1). The common key K_{ab} shared by Users A and B is computed as

$$K_{ab} = \alpha^{K_a K_b} \bmod p = y_b^{K_a} \bmod p = y_a^{K_b} \bmod p. \tag{3.2}$$

That is, User A can recompute the common key K_{ab} by using his secret key K_a and User B's public key y_b. Similarly, User B can retain K_{ab} by using his secret key K_b and User A's public key y_a. This shared common key K_{ab} offers authentication capability between Users A and B.

3.2. Our Scheme

In this section, we present an authentication-combined access control scheme by using a one-way function. Let there be m users and n files in the information system. Let a_{ij} be the access privilege of User i to File j. Initially, by the Diffie-Hellman public key distribution scheme, the system and the users are assigned distinct secret keys and the corresponding public keys, respectively. Let K_s be the secret key and y_s be the public key of the system, and let K_i be the secret key and y_i be the public key of User i. Notice that the users' secret keys are kept by themselves. Let q be a number greater than the maximum value of the a_{ij}'s. The procedure for establishing the information protection system is stated as follows.

PROCEDURE **Establish**.

Step 1. Compute the common key K_{si} shared by the system and User i as

$$K_{si} = y_i^{K_s} \bmod p, \qquad \text{for } i = 1, 2, \ldots, m. \tag{3.3}$$

Step 2. Compute

$$r_{ij} = ((K_{si} + j) \bmod q) \oplus a_{ij}, \qquad \text{for } i = 1, 2, \ldots, m \text{ and } j = 1, 2, \ldots, n, \tag{3.4}$$

where $\oplus$ is the exclusive-or operator.

Step 3. Put the y_i's and r_{ij}'s into a public information table.

The main purpose of procedure **Establish** is to construct common keys shared by the system and the users, respectively. Meanwhile, the access privileges are scrambled by the common keys. After performing procedure **Establish**, the system manages the public information table as shown in Figure 2.

Users (i, y_i) \ j	1	2	⋯	n
(1, y_1)	r_{11}	r_{12}	⋯	r_{1n}
(2, y_2)	r_{21}	r_{22}	⋯	r_{2n}
⋮			⋮	
(m, y_m)	r_{m1}	r_{m2}	⋯	r_{mn}

Figure 2. The public information table managed by the system.

Once the public information table has established, the User i can present his secret key K_i and request the system for accessing the intended File j with the privilege a_{ij}^*. The procedure for verifying the user's request is given as below.

PROCEDURE **Verify** (K_i, i, j, a_{ij}^*).

Step 1. Recompute the common key

$$K_{si} = y_i^{K_s} \bmod p. \tag{3.5}$$

Step 2. Authenticate the requesting user by checking if

$$K_{si} = y_s^{K_i} \bmod p. \tag{3.6}$$

If it is false, then reject the request and terminate.

Step 3. Compute the access privilege as

$$a_{ij} = ((K_{si} + j) \bmod q) \oplus r_{ij}. \tag{3.7}$$

Step 4. Check if a_{ij}^* matches a_{ij}.

If it is true, then accept the request; otherwise reject the request.

With the property of exclusive-or operation, (3.7) is obviously derived from (3.4). By the shared common key, the procedure **Verify** is used not only for computing the access privilege, but also for excluding intruders trying to illegitimately access the protected files. We show how procedure **Verify** can achieve authentication capability. If an intruder pretends to be User i and requests the system for accessing certain protected file, he must first present the correct secret key K_i to pass the test in Step 1 of procedure **Verify**. Again, by the Diffie-Hellman public key distribution scheme, computing K_i from y_i is based on the difficulty of computing a discrete logarithm in GF(p). Thus, procedures **Establish** and **Verify** form a one-way function such that $r_{ij} = F(K_i, y_s, a_{ij})$ and $a_{ij} = F(K_s, y_i, r_{ij})$.

The following example illustrates how procedures **Establish** and **Verify** work.

EXAMPLE 3.1. Consider an information system with the access control matrix shown in Figure 1. Let $\alpha = 2$, $p = 19$ and $q = 5$. Initially, let $K_s = 4$, $y_s = 16$, $K_1 = 2$, $y_1 = 4$, $K_2 = 3$, $y_2 = 8$, $K_3 = 5$, $y_3 = 13$, $K_4 = 7$ and $y_4 = 14$. From (3.2), we have $K_{s1} = 9$, $K_{s2} = 11$, $K_{s3} = 17$ and $K_{s4} = 17$. After performing procedure **Establish**, we have the public information table shown in Figure 3.

Users (i, y_i) \ j	Files 1	2	3	4	5
(1, 4)	4	5	3	1	4
(2, 8)	0	1	5	0	2
(3, 13)	3	5	4	2	1
(4, 14)	2	6	0	1	6

Figure 3. The public information table for Example 3.1.

From Figure 3, by (3.7), we compute

$$\begin{aligned} a_{12} &= (((y_1^{K_s} \bmod p) + 2) \bmod q) \oplus r_{12} = ((K_{s1} + 2) \bmod 5) \oplus r_{12} \\ &= ((9 + 2) \bmod 5) \oplus 5 = 1 \oplus 5 \\ &= 4, \qquad \text{which is correct.} \end{aligned}$$

Again,

$$\begin{aligned} a_{24} &= (((y_2^{K_s} \bmod p) + 4) \bmod q) \oplus r_{24} = ((K_{s2} + 4) \bmod 5) \oplus r_{24} \\ &= ((11 + 4) \bmod 5) \oplus 0 = 0 \oplus 0 \\ &= 0, \qquad \text{which is correct.} \end{aligned}$$

The reader may verify the other privileges by using (3.7).

Now, consider the access control of our scheme in dynamic environments, such as change access privileges and insert/delete users or files. It is an easy task to change access privileges and delete users or files. To change the access privilege of User i to File j to a_{ij}^*, we only compute

$$r_{ij}^* = (((y_i^{K_s} \bmod p) + j) \bmod q) \oplus a_{ij}^*,$$

and update the entry (i, j) of the public information table to r_{ij}^*. To delete User t from the system, we only eliminate y_t and r_{tj}'s, for $j = 1, 2, \ldots, n$, from the public information table. Similarly, we only eliminate r_{it}'s, for $i = 1, 2, \ldots, m$, from the public information table for deleting File t.

To insert a File t into the system, we compute

$$r_{it} = (((y_i^{K_s} \bmod p) + t) \bmod q) \oplus a_{it}, \qquad \text{for } i = 1, 2, \ldots, m,$$

and add the r_{it}'s to the public information table. As to insert a User t into the system, we first assign a distinct secret key K_t and its corresponding public key y_t to User t. Then, we compute

$$r_{tj}^* = (((y_t^{K_s} \bmod p) + j) \bmod q) \oplus a_{tj}, \qquad \text{for } j = 1, 2, \ldots, n,$$

and add the r_{tj}'s to the public information table. For the consideration of security, the newly assigned secret key K_t should not be previously used in key distribution. A pseudo-random number generator with large period can be applied for this purpose [11]. The following example shows the dynamic access control of our scheme.

EXAMPLE 3.2. Reconsider the access control matrix in Figure 1. Figure 4 shows the public information table after deleting User 2 and File 3. Figure 5 shows the results of inserting File 6. For inserting File 5, we first assign $K_5 = 6$ and $y_5 = 7$. The results of inserting User 5 are shown in Figure 6.

Users (i, y_i) \ j	Files 1	2	4	5
(1, 4)	4	5	1	4
(3, 13)	3	5	2	1
(4, 14)	2	6	1	6

Figure 4. The public information table after deleting User 2 and File 3.

Users i \ j	1	2	3	4	5	6
1	4	4	1	2	0	2
2	2	2	1	0	3	1
3	0	1	4	3	3	4
4	1	2	0	0	4	0

(a) Access control matrix.

Users (i, y_i) \ j	1	2	3	4	5	6
(1, 4)	4	5	3	1	4	2
(2, 8)	0	1	5	0	2	3
(3, 13)	3	5	4	2	1	7
(4, 14)	2	6	0	1	6	3

(b) Public information table.

Figure 5. The results of inserting File 6.

Users i \ j	1	2	3	4	5
1	4	4	1	2	0
2	2	2	1	0	3
3	0	1	4	3	3
4	1	2	0	0	4
5	2	3	1	0	2

(a) Access control matrix.

Users (i, y_i) \ j	1	2	3	4	5
(1, 4)	4	5	3	1	4
(2, 8)	0	1	5	0	2
(3, 13)	3	5	4	2	1
(4, 14)	2	6	0	1	6
(5, 7)	1	7	1	1	0

(b) Public information table.

Figure 6. The results of inserting User 5.

4. CONCLUSIONS

We have presented an authentication-combined access control scheme by using a one-way function. By the Diffie-Hellman public key distribution scheme, the system and the users are initially assigned distinct secret keys and the corresponding public keys, respectively. The users' secret keys are kept by themselves. When request the access to an intended file, the user should present his secret key to the system. Being different from the previously proposed schemes, our scheme is safer, and the presented user's secret key is used not only for computing the access privilege but also for authenticating the user himself.

As to the establishment of our access control system, we see that it needs only $\lceil \log K_s \rceil$ multiplications, one addition and two modular operations plus one exclusive-or operation. For verifying an access request, it requires $\lceil \log K_i \rceil + \lceil \log K_s \rceil$ multiplications, one addition, two modular operations, two comparisons plus one exclusive-or operation. Besides, our proposed scheme can perform the access control in dynamic environments without affecting any user's secret key.

REFERENCES

1. D.E.R. Denning, *Cryptography and Data Security*, Addison-Wesley, Readings, MA, (1982).
2. G.S. Graham and P.J. Denning, Protection—Principals and practices, *Proceedings AFIPS 1972 SJCC*, pp. 417–429, AFIPS Press, Montvale, NJ, (1972).
3. J.H. Saltze and M.D. Schroeder, The protection of information in computer system, *Proceedings of IEEE* **63**, 1278–1308 (1975).
4. C.C. Chang, On the design of a key-lock-pair mechanism in information protection systems, *BIT* **26** (4), 410–417 (1986).
5. C.C. Chang, An information protection scheme based upon number theory, *The Computer Journal* **30** (3), 249–253 (1987).
6. C.C. Chang and J.K. Jan, An access control scheme for new users and files, *The International Journal of Policy and Information* **12** (2), 89–98 (1988).
7. C.K. Chang and T.M. Jiang, A binary single-key-lock system for access control, *IEEE Trans. on Computers* **C-38** (10), 1462–1466 (1989).
8. J.K. Jan, A single key access control scheme in information systems, *Information Science* **51** (1), 1–11 (1990).

9. C.S. Laih, L. Harn and J.Y. Lee, On the design of single-key-lock mechanism based on Newton's interpolating polynomials, *IEEE Trans. on Software Engineering* **SE-15** (9), 1135–1137 (1989).
10. M.L. Wu and T.Y. Hwang, Access control with single-key-lock, *IEEE Trans. on Software Engineering* **SE-10** (2), 185–191 (1984).
11. D.E. Knuth, *The Art of Computer Programming, Volume 2, Semi-numerical Algorithms*, 2nd edition, Addison-Wesley, Reading, MA, (1981).
12. H.C. Williams, Computationally 'hard' problems as a source for cryptosystems, In *Secure Communications and Asymmetric Cryptosystems, AAAS Selected Symposium 69* (Edited by G.J. Simmons), pp. 11–39, Westview Press, Colorado, (1982).
13. W. Diffie and M.E. Hellman, New directions in cryptography, *IEEE Trans. on Information Theory* **IT-22** (6), 644–654 (1976).
14. S.C. Pohlig and M.E. Hellman, An improved algorithm for computing logarithms over GF(p) and its cryptographic significance, *IEEE Trans. on Information Theory* **IT-24** (1), 106–110 (1978).